Amendments to the Claims

The listing of claims will replace all prior versions, and listings of claims in the application.

1. (Previously Presented) An apparatus comprising:

a processor capable of simultaneous execution of two or more threads of instructions, wherein the processor comprises:

at least two resource units, each capable of being assigned to an instruction of each of the threads, wherein each of the resource units implements a portion of instruction types occurring in each of the threads;

a priority register to store thread information for the threads, the thread information including a priority code corresponding to the instructions of each thread, at least one of the threads requesting use of one of the resource units for processing a current instruction; and

a priority selector coupled to the priority register to generate an assignment signal to assign the resource units to the requesting thread's current instruction according to the priority codes.

2. (Previously Presented) The apparatus of claim 1 wherein each of the resource units is one of an instruction fetch unit, a memory locking unit, a load unit, a store unit, an input/output unit, a peripheral unit interface, and a functional unit.

3. (Original) The apparatus of claim 2 wherein the functional unit is one of an arithmetic unit, a logic unit, and an arithmetic and logic unit.

4. (Original) The apparatus of claim 1 further comprising:

an instruction multiplexer coupled to the priority selector to pass instructions stored in a plurality of instruction registers to execution units according to the assignment signal.

5. (Previously Presented) The apparatus of claim 1 further comprising:

a priority assignor coupled to the priority register to set the thread information including at least one of the priority codes corresponding to the instructions of each thread in response to a start instruction from an instruction decoder and dispatcher.

6. (Previously Presented) The apparatus of claim 5 wherein the priority assignor sets an active flag in the priority register corresponding to at least one of the threads in response to the start instruction.

7. (Previously Presented) The apparatus of claim 6 wherein the priority assignor resets the active flag in the priority register corresponding to at least one of the threads in response to a quit instruction from the instruction decoder and dispatcher.

8. (Previously Presented) The apparatus of claim 1 wherein the priority selector assigns one of the resource units to the instructions of one of the threads if the one of the threads is not served and the one of the resource units is free.

9. (Previously Presented) The apparatus of claim 8 wherein the one of the threads has highest priority code among a set of ready threads.

10. (Previously Presented) The apparatus of claim 8 wherein the priority selector iteratively assigns resource units to instructions of threads in the set of ready threads according to the corresponding priority codes and resource availability until the set becomes empty.

11. (Previously Presented) A method comprising:

executing two or more threads of instructions simultaneously in a processor;

storing thread information for the threads, the thread information including a priority code corresponding to instructions of each thread, at least one of the threads requesting use of one of at least two resource units capable of being assigned to an instruction of each of the threads for processing a current instruction, wherein each of the resource units implements a portion of instruction types occurring in each of the threads; and

generating an assignment signal to assign the resource units to the requesting thread's current instruction according to the priority codes.

12. (Previously Presented) The method of claim 11 wherein each of the resource units is one of an instruction fetch unit, a memory locking unit, a load unit, a store unit, an input/output unit, a peripheral unit interface, and a functional unit.

13. (Original) The method of claim 12 wherein the functional unit is one of an arithmetic unit, a logic unit, and an arithmetic and logic unit.

14. (Original) The method of claim 11 further comprising:

passing instructions stored in a plurality of instruction registers to execution units according to the assignment signal.

15. (Previously Presented) The method of claim 11 further comprising:

setting the thread information including at least one of the priority codes corresponding to the instructions of each thread in response to a start instruction from an instruction decoder and dispatcher.

16. (Previously Presented) The method of claim 15 wherein setting the thread information comprises setting an active flag in the priority register corresponding to at least one of the threads in response to the start instruction.

17. (Previously Presented) The method of claim 16 wherein setting the thread information comprises resetting the active flag in the priority register corresponding to at

least one of the threads in response to a quit instruction from the instruction decoder and dispatcher.

18. (Previously Presented) The method of claim 11 wherein generating the assignment signal comprises generating the assignment signal to assign one of the resource units to the instructions of one of the threads if the one of the threads is not served and the one of the resource units is free.

19. (Previously Presented) The method of claim 18 wherein the one of the threads has highest priority code among a set of ready threads.

20. (Previously Presented) The method of claim 1 wherein generating the assignment signal comprises iteratively assigning resource units to instructions of threads in the set of ready threads according to the corresponding priority codes and resource availability until the set becomes empty.

21. (Previously Presented) A processor capable of simultaneous execution of two or more threads of instructions, comprising:

at least two resource units to provide resources for use by the threads, each capable of being assigned to an instruction of each of the threads, wherein each of the resource units implements a portion of instruction types occurring in each of the threads;

a resource prioritizer coupled to each of the resource units to prioritize resource utilization, the resource prioritizer comprising:

a priority register to store thread information for the threads, the thread information including a priority code corresponding to the instructions of each thread, at least one of the threads requesting use of one of the resource units for processing a current instruction, and

a priority selector coupled to the priority register to generate an assignment signal to assign the resource units to the requesting thread's current instruction according to the priority codes.

22. (Previously Presented) The processor of claim 21 wherein each of the resource units is one of an instruction fetch unit, a memory locking unit, a load unit, a store unit, an input/output unit, a peripheral unit interface, and a functional unit.

23. (Original) The processor of claim 22 wherein the functional unit is one of an arithmetic unit, a logic unit, and an arithmetic and logic unit.

24. (Original) The processor of claim 21 the resource prioritizer further comprising:

an instruction multiplexer coupled to the priority selector to pass instructions stored in a plurality of instruction registers to execution units according to the assignment signal.

25. (Previously Presented) The processor of claim 21 the resource prioritizer further comprising:

a priority assignor coupled to the priority register to set the thread information including at least one of the priority codes corresponding to the instructions of each thread in response to a start instruction from an instruction decoder and dispatcher.

26. (Previously Presented) The processor of claim 25 wherein the priority assignor sets an active flag in the priority register corresponding to at least one of the threads in response to the start instruction.

27. (Previously Presented) The processor of claim 26 wherein the priority assignor resets the active flag in the priority register corresponding to at least one of the threads in response to a quit instruction from the instruction decoder and dispatcher.

28. (Previously Presented) The processor of claim 21 wherein the priority selector assigns one of the resource units to the instructions of one of the threads if the one of the threads is not served and the one of the resource units is free.

29. (Previously Presented) The processor of claim 28 wherein the one of the threads has highest priority code among a set of ready threads.

30. (Previously Presented) The processor of claim 28 wherein the priority selector iteratively assigns resource units to instructions of threads in the set of ready threads according to the corresponding priority codes and resource availability until the set becomes empty.